Charles River Analytics

Final Report No. R07130-2
Charles River Analytics Contract No. C07130
Government Contract No. W31P4Q-08-C-0126
Expiration Date of SBIR Data Rights Period: 14 August 2013

Modular Affective Reasoning-based Versatile Introspective Architecture (MARVIN) Phase I Final Report

Scott Neal Reilly, Sean Guarino and Mike Reposa

Charles River Analytics
625 Mount Auburn Street
Cambridge, MA 02138

14 August 2008

This material is based upon work sponsored by the Defense Advanced Research Projects Agency, DARPA Order AL82-51, issued by the U.S. Army Aviation and Missile Command under Contract No. W3P4Q-08-C-0126. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US Army Aviation and Missile Command.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
08-04-2008	Final Report	12/20/2007 – 8/14/2008

4. TITLE AND SUBTITLE
Modular Affective Reasoning-based Versatile Introspective Architecture (MARVIN)

5a. CONTRACT NUMBER
W31P4Q-08-C-0126

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Scott Neal Reilly, Sean Guarino and Mike Reposa

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Charles River Analytics Inc.
625 Mount Auburn Street
Cambridge, MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER
SBIR Topic #: SB072-009
DoD Proposal #: D072-009-0055

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency/IPTO
3701 North Fairfax Drive
Arlington, VA 22203-1714

10. SPONSOR/MONITOR'S ACRONYM(S)
DARPA/IPTO

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Report developed under SBIR Contract for Topic # SB072-009. In this Phase I study, we demonstrated that affect-inspired, self-aware mechanisms can significantly improve the performance of computational systems in cases of limited resources. We designed four affect-inspired adaptation mechanisms and argue that all of these mechanisms have the ability to positively impact the performance of various types of computational systems. We developed a MINIX-based implementation of an affective attention and resource management module. For evaluation and demonstration, we developed a UAV-based scenario that requires the UAV to respond rapidly to environmental events. We found that the affective system dramatically outperformed a comparable non-affective system. This was not, however, at the expense of non-critical operations as the critical response processes were able to dynamically reduce their resource usage in non-critical periods, enabling the other processes to be even more efficient than their non-affective counterparts.

15. SUBJECT TERMS
SBIR Report, Self-Aware Computing, Affective Computing, Affective Reasoning, Metacognition, Adaptive Processing, Reactive Systems, Self-Managing Systems, Self-Optimizing Processing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Unlimited	26	Dr. Scott Neal Reilly
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER *(Include area code)* 617-491-3474 x542

Acknowledgment

This effort was sponsored by the Defense Advanced Research Projects Agency, DARPA Order AL82-51, issued by the U.S. Army Aviation and Missile Command under Contract No. W3P4Q-08-C-0126. The authors would like to thank Dr. Bill Harrod, Jon Hiller, and Kat Seifert for their support and guidance throughout this effort.

Table of Contents

List of Figures

Glossary of Abbreviations

APMM	Affective Process Management Module
ATD	Advanced Technology Demonstration
C2	Command and Control
CPU	Central Processing Unit
FCS	Future Combat System
FFW	Future Force Warrior
FWTI	Future Warrior Technology Integration
MARVIN	Modular Affective Reasoning-Based Versatile Introspective Architecture
MCL	Meta-Cognitive Loop
OS	Operating System
PDA	Personal Digital Assistant
UAV	Unmanned Aerial Vehicle
SBIR	Small Business Innovative Research

1. Introduction

1.1 Problem Description

Robotic vehicles, unmanned systems, critical command and control (C2) systems, and other warfighter-critical computational systems, such as those being developed by the U.S. Army's Future Warrior Integration Team (US Army, 2006) (FWTI) program for integration into the Future Combat System (FCS), all rely on computational systems that need to operate effectively in complex, dynamic, and adversarial environments. Currently fielded systems are not up to par, since they can be immobilized by simple component failures. Future operations will not tolerate this level of brittleness, especially in complex, autonomic systems, such as unmanned vehicles, where human corrections to the system may be limited. In addition, systems will need to deal with infrastructure failures (in both hardware and software) that result from a variety of causes from simple design errors to an adversary's cyberwarfare attack against network and processing vulnerabilities. These issues become even more challenging in limited hardware environments, such as those used on unmanned vehicles and those used by Ground Soldiers, which often involve computational restrictions in terms of power, battery life, CPU speed, and network bandwidth.

Unlike the dynamic environments in which they operate, modern operating systems (including, for the purposes of this proposal, operating systems responsible for process and task management, resource allocation, and network management) are relatively static in nature, with full capabilities determined well *before* actual deployment. When hardware capabilities, available resources, or mission requirements change, current systems are limited to their original mode of operation, resulting in inefficient resource use, performance penalties, and, at worst, catastrophic failure. For example, in existing multi-threaded systems, approaches to using system resources (e.g., processor time, network bandwidth) for multiple tasks are preset, creating potential difficulties in supporting dynamic requirements that can lead to a loss of high priority activities and mission failure, despite the availability of alternative resources. To prevent such failures, systems throughout the military need more dynamic processing capabilities that can use resources more effectively to meet dynamic requirements in unforgiving or adversarial environments.

Humans and other biological creatures, while far from perfect, are considerably better at adapting to dynamic environments than are computational systems. Therefore, the study of human adaptation might provide insights into mechanisms for improving the performance of static, computational systems. While it might not be obvious at first why adaptation mechanisms that work for humans should be suitable for computational systems, we believe that there are

enough architectural similarities that such a transfer of effective adaptation mechanisms seems plausible. For instance, they both have multiple, concurrent threads/goals competing for time/attention resources, they are both limited in terms of resources, time to respond, and ability to perceive of the environment, they both have limited short & long term memory, and both act in a social/networked environment.

Self-Aware Computing uses this approach to provide a conceptual, *metacognitive* approach for generating dynamic capabilities for computational systems inspired by the adaptive, introspective capabilities of organic/biological systems (Agarwal & Harrod, 2006; Ganek & Corbi, 2003). These metacognitive approaches are designed to observe and adapt their own processing, as well as other processing within the self-aware system, in an effort to achieve their specified goals even in dynamic environments. Agarwal & Harrod (2006) identify a number of desirable properties for self-aware systems, each of which is associated with the key goal of making the processing system adaptive. In addition to general adaptation capabilities, the systems must be self-aware, observing themselves and their behavior; they must be goal-oriented to accomplish processor goals, and influenced by both user and application goals; they must be resilient, able to take corrective actions in the face of potential faults and threats (e.g., self-healing and, by extension, self-protective in network-centric environments (Ganek et al., 2003)); and they must perform approximate reasoning, adapting the assignment of computational resources based on current needs.

Current metacognitive approaches concentrate on resiliency issues, including the OceanStore effort, which provides durable long-term storage utilities for billions of network users (Rhea et al., 2003), and the Recovery-Oriented Computing project, which is exploring fault diagnosis and resolution by taking an approach that embraces inevitable failures, detecting them as they happen and resolving them after that point, rather than preventing them (Patterson et al., 2002). Other research concentrates on more generic self-aware concepts, such as University of Maryland's Meta-Cognitive Loop (MCL)-based systems, which monitors events, assesses strategies to deal with them, and utilizes strategies without interruption to monitoring (Anderson & Perlis, 2005). In addition to each of these research-oriented efforts, IBM has made autonomic computing a key component of their current and future development of adaptive systems.

None of these traditional approaches, however, makes any effort to utilize the rich conceptual resources available in human *affective* reasoning-based adaptation. We believe that this particular niche will dramatically assist in moving self-aware processing forward. While many think of emotions and moods as being irrational, psychologists and neuroscientists have largely come to believe that affect is an evolutionarily adaptive aspect of human behavior that is useful for living in dynamic, resource-bounded, dangerous, social environments (Damasio, 1994). Computational

scientists and philosophers have come to believe that computational systems with multiple, competing motivations, operating with limited resources (including time, memory, and computational power), and operating in complex, dynamic, and social environments will *require* affect-like mechanisms to be effective (Minsky, 2006; Pfeifer, 1993; Sloman & Croucher, 1981; Toda, 1962).

1.2 Technical Approach

This effort is inspired by the following insights:

- *Humans and biological entities are not perfect, but are much better than current computational systems at adapting to dynamic environments.*
- *They share enough key architectural features that it is plausible that understanding how humans adapt can suggest mechanisms that would help computational systems adapt.*
- *Affect is a human adaptation mechanism that has the potential to suggest/inspire computational adaptation mechanisms.*

The goal of this effort has been to build on these insights and to explore the plausibility of using mechanisms inspired by human affect to improve the robustness and effectiveness of computation systems used by the U.S. military and others. To do this, we pursued an effort consisting of the following tasks:

Task 1: Define Requirements and Scope. The objective of this task was to determine the types of computational systems that might be suitable for affect-based adaptation mechanisms and to lay out a basic approach, both for eventual transition and short-term, rapid prototyping and evaluation.

Task 2: Perform Human Affect and Affective Modeling Literature Review. The objective of this task was to identify affect-based adaptation mechanisms that might improve the performance of computational systems.

Task 3: Design Affective Reasoning Modules. The objective of this task was to refine the design of a number of concepts for affective reasoning modules to the point where initial prototype implementations could be made.

Task 4: Develop Demonstration Prototype. The objective of this task was to implement a prototype version of one or more of the affective reasoning modules designed under Task 3.

Task 5: Develop Evaluation Metrics. The objective of this task was develop suitable metrics for evaluating self-aware systems that could be applied in Phase II or, as possible, to the prototype developed under Task 4.

Task 6: Write Final Report. The objective of this task was to generate this report, which summarizes our approach, Phase I results, and recommended future directions.

Task 7: Investigate Commercialization Opportunities. The objective of this task was to identify existing and developing systems that could benefit from affective adaptation mechanisms, including both commercial systems and military systems.

1.3 Summary of Results

First, we identified a number of types of computational systems that might be suitable for affect-based adaptation mechanisms, including systems operating in adversarial, time- and bandwidth- and other-resource-constrained environments, performing multiple functions, and interacting with other computational systems. In particular, we believe UAVs and Ground Soldier PDAs are excellent examples of existing, operational systems that have the necessary properties to significantly benefit from affect-inspired adaptations. We also laid out a basic development approach, that starts in the Phase I with MINIX, an easy-to-modify version of Linux, and simulated UAVs and that moves, in Phase II, to Linux and operational systems, including UAVs and Ground Soldier systems.

Second, we had planned to perform a literature search to identify additional affect-based adaptation mechanisms. However, based on feedback from the Sponsor, we scaled back on this effort in favor of concentrating on a thorough exploration of a single affective computing concept. The four ideas we had initially developed, based on our experience in the area and familiarity with the literature, were deemed more than sufficient for Phase I. We did, however, identify additional sources in the existing affect literature, including (Nussbaum, 2001; Preifer, 1993; Pfeifer, 1998; Sloman et al., 1981; Toda, 1962; Toda, 1982; Lazarus, 1991; Frijda, 2006), which we believe can easily be mined for additional affect-inspired ideas in the future.

Third, we developed four concepts for affect-inspired mechanisms to improve the performance of computational systems: (1) an affective attention and resource management module, (2) a generic, multi-tiered affective adaptation module, (3) an emergency resource-allocation module, and (4) a social-affect module. Based on feedback from the Sponsor at the kickoff briefing, instead of focusing broadly on fully designing these four concepts, we further concentrated on the first of these concepts and designed it to a level where it was ready to be implemented as a testable prototype system.

Fourth, we implemented an affect-inspired resource allocation mechanism that we call the Affective Process Management Module (APMM) on the MINIX operating system. We also developed a UAV scenario, including a set of tasks, an implementation of both standard and affective processes to simulate those tasks, and a script of events to drive the evaluation

simulation. By developing both a standard MINIX version and an affective version of these processes, we were able to compare the performance of an APMM-enhanced OS implementation to that of a traditional MINIX OS.

Fifth, we investigated suitable metrics for self-aware affective systems. Again, following consultation with the Sponsor, we concentrated on metrics for evaluating the APMM, rather than on developing generic metrics usable for any affective OS component. We designed an evaluation using the implementation of the APMM and the UAV scenario developed under Task 4. We found that the APMM performed better in time-critical situations by allocating more CPU time to critical event handlers in those situations. We also found that the APMM performed better in non-time-critical situations as the emergency event handlers were able to be provided fewer resources, enabling improved performance of the non-critical processes. We believe similar types of evaluation metrics can be developed for other affect-based mechanisms and domains.

Sixth, we generated this report, which describes our approach, Phase I results, and recommendations for follow-on work.

Seventh, we pursued a number of possible transition paths, focusing primarily on military systems, though also brainstorming commercial applications that can be pursued further in a follow-on Phase II now that there is a working demonstration system. The military systems of interest generally include those that operate in resource-limited situations, such as UAVs and Ground Soldier systems, where adaptive OS elements can be more essential to system stability. We pursued initial discussions with large system integrators that build UAV systems and Ground Soldier systems and have gotten interest in further discussing transition of MARVIN technologies to these systems in Phase II and Phase III.

Based on the results of this effort, we have demonstrated the substantial, achievable benefits available to the next generation of adaptive computational systems by leveraging ideas from human affect. We believe these results provide the foundation for additional research and development that focuses on designing and developing additional affect-based computational models, rigorously testing and evaluating the performance and resilience improvements provided by affect-based modules, transitioning the computational improvements to a Linux deployment environment in order to further evaluate the applicability of MARVIN mechanisms to more widely deployed and rich operating systems and to prepare it for further testing and deployment on real-world systems, and continuing to explore government-transition and commercialization opportunities.

2. Review of Phase I Objectives and Results

2.1 Phase I Technical Objectives

The primary objective of this Phase I effort was to design and demonstrate the feasibility of the Modular Affective Reasoning-Based Versatile Introspective Architecture (MARVIN), a novel architecture for developing components to enhance self-aware computational systems by exploiting human affective processing mechanisms. Specific objectives included answering the following questions:

- **Define System Objectives**. What are the specific aspects of computational systems that require self-awareness? How do we select a subset of these within a demonstration scenario to match the scope of a Phase I effort? What are the primary technical requirements and constraints that MARVIN must meet? What are the most appropriate uses of MARVIN in providing system adaptability and maintaining system integrity?
- **Investigate Human Affect and Affective Modeling Approaches**. What aspects of the human affective system can best be utilized in military self-aware systems, and how can we best model those aspects? How can we use affect to inspire enhancements to cognitive models to provide the short-term, mid-term, and long-term adaptation required in future military systems? How can concepts from affective social interactions and emotional contagion enhance network and process communications?
- **Design Affective Reasoning Modules**. How can we best utilize affective models in designing modules for self-aware systems? How can we design a modular affective system to be integrated with existing metacognitive self-aware systems? What technologies are most applicable in the design and development of these modules? What are the best aspects of resource, process, and network management to moderate with affect, and how can we best apply affective principles to those management processes?
- **Demonstration**. What are the requirements of a virtual computing environment that can effectively demonstrate the capabilities of affective reasoning modules within an appropriate demonstration scenario? What are the requirements for an affective protocol manager that can integrate with other self-aware systems, and how can we implement those requirements within the selected virtual computing environment? How can we best make this virtual system compelling for the Sponsor-established scenario?
- **Develop Evaluation Metrics**. What existing OS and processor metrics are useful for analyzing affective self-aware systems, and what new metrics are required to analyze their key features (e.g., self-awareness, resilience, goal-orientation, approximate reasoning, and adaptation capabilities)? How can these metrics be updated to analyze the improvements that

affective reasoning modules are expected to produce? How can these metrics be utilized by our affective reasoning modules?

- **Identify and Evaluate Commercialization and Transition Opportunities**. What are the most suitable transition partners for affective self-aware systems within the U.S. government? What is an appropriate plan for transitioning to such partners? What are suitable commercial markets for affective self-aware processing technology?

The successful completion of these objectives forms a sound basis for the advanced development and thorough evaluation of a full-scope prototype in Phase II.

2.2 Phase I Results

In this section, we review the tasks that we performed in pursuit of the objectives listed in the previous section and the corresponding results.

Task 1: Define Requirements and Scope. In conjunction with the Sponsor at the kickoff briefing, we decided to focus only minimally on the design of a wide range of affect-inspired adaptation mechanisms and put additional time into the more complete design and implementation of a single demonstrable mechanism. We still performed a limited review of the affect literature, brainstormed a number of potentially useful affect-inspired adaptation mechanisms, and identified additional literature to be used to extend and refine this list under a potential follow-on Phase II effort. But we spent more effort on the design, implementation, and evaluation of a prototype Affective Process Management Module (APMM) that demonstrates the basic benefits and feasibility of affect-inspired reasoning in OS operations and provides a basis for the development of further affective modules.

In terms of platform types and transition targets, we were told there were no specific requirements. Therefore, we explored the use of these technologies in primarily unmanned vehicles, such as mini and micro UAVs, and Soldier-based systems, such as the Army's Future Warrior Technology Integration (FWTI) Team's PDA-based Basic Soldier system and the Thermite- and laptop-based Leader Systems. FWTI is follow-on from the earlier Future Force Warrior (FFW) Advanced Technology Demonstration (ATD). Because of the resource limitations (memory, CPU, network, battery) and time limitations placed on these systems by their use in dynamic, dangerous environments, we believe they are well suited as target platforms. In addition, our participation as a member of the FWIT under a parallel SBIR effort provides us access to the developers, a number of end users, the platform requirements, the platforms, and other contractors that are working on relevant problems. Additionally, we identified MINIX as a suitable prototype development platform for our Phase I effort, effective

at supporting rapid implementation and testing of affective OS components without the complexity of implementing those components within a larger OS, such as Linux.

Task 2: Perform Human Affect and Affective Modeling Literature Review. Based on feedback from the Sponsor at the kickoff briefing, we scaled back our extended literature review, which had been intended to identify a broad range of possible affect-inspired mechanisms for improving the robustness and effectiveness of computational systems, in favor of concentrating on implementing a single mechanism in some depth to prove the viability of the concept of affective OS modules. Nevertheless, we identified a number of sources that we believe would be worth exploring for additional ideas under a potential follow-on Phase II effort, including (Nussbaum, 2001; Preifer, 1993; Pfeifer, 1998; Sloman et al., 1981; Toda, 1962; Toda, 1982; Lazarus, 1991; Frijda, 2006). Additionally, we were able to identify a number of affect-inspired improvements based on our existing knowledge of the affect literature. In particular, we identified and developed initial ideas for four affect-inspired mechanisms that we believe have the potential to improve the performance of modern computational systems:

Affective attention and resource management. Human affect moderates attentional and other resources (e.g., physical resources via hormones) based on the state of the current goals and the relevant aspects of the environment. We identified two initial mechanisms based on this concept, one related to fear and the other to hope, that we have used to inspire computational adaptation mechanisms.

First, as important goals are threatened, the system must adapt to protect them by allocating additional needed resources. This is comparable to human fear. Fear results in attentional/perceptual focusing (also called *tunneling* in the psychology literature) that filters out elements of the environment that would normally be attended to by resources that are better allocated to the threatened goal. Fear similarly affects physical resources. For instance, when threatened, humans will release adrenaline that causes them to be able to react physically more rapidly and powerfully. However, they would not want to maintain this level of physical readiness continuously, as it would lead to exhaustion.

We believe that there are a number of computational systems that can benefit from such a mechanism. For instance, as we will discuss further below, operating systems that are responsible for allocating resources (CPU, memory, cache, network access) to processes can more effectively allocate these resources if they are aware when the processes are being threatened. For instance, a key UAV navigation process is threatened when the UAV is in danger of colliding with another object but does not normally require large number of resources for normal flight. From a physical standpoint, Soldier-based systems such as the FWTI ensemble have very limited battery power. The ability to allocate battery power to overcome short-term

threats but to otherwise reserve power could dramatically increase the effectiveness of such systems. Obviously, the success of such mechanisms relies on a level of self-awareness on the part of the processes to recognize threats, but one of the results of this effort was to demonstrate that it is possible to create such self-aware processes.

Second, hope occurs when a goal is anticipated to succeed. In humans, hope does not play as strong a role as fear, but it does play a role in focusing attentional and physical resources. As goal success is neared, the goal will get additional resources to ensure its completion. While we believe the payoff for hope-inspired mechanisms is not as large as for fear, there are likely still ways to take advantage of such a mechanism. For instance, in operating systems, context switching is a significant cost and reducing context switches has the potential to improve overall efficiency. By noticing when processes are nearing completion, the operating system can allocate additional CPU time to complete them without switching (or with minimal switching) to other processes.

This concept formed the basis for the prototype affect module that we designed and implemented under this Phase I effort and it is described in more detail under the following task descriptions.

Generic, multi-tiered affective adaptation. Affect helps humans adapt on a number of time scales. Emotions provide rapid, short-term, and relatively specific adaptation to the environment. Moods provide longer-term, more generic adaptation to an environment. Attitudes (or personality traits) provide much longer term and more generic adaptation for the human in all situations. In addition, these are interrelated as, for instance, attitudes influence an individual's mood, which, in turn, influence emotions from moment to moment. Similarly, emotions can lead to mood changes which, over long periods, can produce attitude changes. Moods and attitudes continue to effect decision making even beyond the end of the emotion, and thus provide an extended adaptation of behavior beyond the end of an initial reaction.

The result is a general-purpose adaptation mechanism that changes behavior and behavior tendencies over time based on successes and failures in the environment. For instance, if a human is threatened, they feel fear (the emotion and, to a lesser extent at first, the mood) and respond based on the intensity of the fear, which is based on a number of factors, including the level of threat and the predisposition towards fear. The emotion subsides quickly and the mood subsides slowly. If a second threat occurs relatively near in time, it results in fear being generated more strongly due to the remaining mood. Eventually, with enough such events, the human's attitude will change, leading to a skittish personality that is always on edge. While, in more stable environments, such a state would be inappropriate, in an environment where true threats are abundant, the skittish personality could be a reasonable adaptation.

This is a generic mechanism in that we are not assuming that the human is learning anything in particular about the threat or suitable responses to it (though, they certainly might do that too), but rather we are assuming a general tendency towards certain kinds of reactions that are generally suited to a particular environment (e.g., a dangerous environment where more caution is warranted). Similar examples can be generated for aggression and other traits as well. For instance, more competitive environments might result in more aggressive traits after a series of conflicts.

We believe that such a set of multi-tiered adaptation mechanisms has applications to computational systems as well. For instance, a system might change its network behavior based on actual traffic patterns, network reliability, and vulnerabilities. As these properties of the environment change over time, the system will be able to adapt to these new states as well. Also, these changes will be temporary in the case of temporary changes in the environment, and more stable in the case of longer-term environmental states.

Emergency resource-allocation mechanisms. Humans have two kinds of affective reaction mechanisms. A conscious mechanism enables response to learned and noticed states and events (e.g., "I got angry when I realized that she was going to..."). This is typically referred to a secondary appraisal mechanism. A reflexive mechanism, called a primary appraisal mechanism, provides rapid, instinctual, subconscious responses to typically dangerous situations (e.g., heights, loud noises). These are critical when there isn't time to think, plan, or reason out the optimal response, but not reacting is almost certainly bad.

We believe that we can build a computational mechanism that mimics aspects of such an emergency-response mechanism. For instance, modern operating systems maintain a queue of processes that are allocated CPU time in a round-robin manner, with additional time being provided to higher-priority threads. Even if we were to implement a fear-based resource-allocation mechanism as described under the previous concept, the threatened thread is still part of the system queue, which means that it might not be identified as being in an emergency state until after cycling through all processes and that it is still sharing significant resources with the other processes. We believe it would be useful to have a second, emergency process queue that takes priority over the standard queue and that includes a very small number of system-critical threads to be executed in cases of emergency. By creating very simple, critical-state identification monitors that can transfer key processes from the standard queue to the emergency queue and then allocate all resources to the processes in that queue for the duration of the emergency (or until system failure), we can dramatically increase the rapidity of the possible response to a system-critical event.

Unlike the previous concept, this approach would only be practical in the most serious of situations, but it could provide a mechanism for more effectively responding to such situations. For instance, we envision such a mechanism being useful in cases of pending system failure (e.g., due to power loss or, in the case of a UAV, a crash) or possibly a serious security breach, where one last-minute effort might either recover the system, get critical data off of the system, or save critical data from a security breach. Existing operating systems cannot react effectively to these kinds of critical system threats. Traditional self-aware systems begin to provide necessary metacognitive analysis (Gelenbe & Loukas, 2007; Mowbray & Bronstein, 2002). However, in many cases, they will fail to react within required time constraints. By incorporating an emergency resource-allocation mechanism that steps outside of the standard resource-allocation processes, self-aware systems can more successfully respond in such system-critical situations.

Social affect. A key role of affect is in rapidly communicating important information with others. Humans are pre-programmed to generate emotional responses that indicate danger, disturbances, and pleasure. In response, we are likewise pre-programmed to respond to these emotional displays with emotional and motivational reactions. For instance, displays of sadness will typically result in pity and the associated helping behaviors in others. Fear displays will often result in fear in others, which is used, for instance, to rapidly spread messages of danger through a herd of animals. In general, these forms of social affect have been shown to make the individuals and the society more successful (Jaffe, 2006), and we anticipate that similar rapid, low-bandwidth messages could help both in the context of networked computers or the context of multiple processes operating within the same process space.

As with the other mechanisms we have discussed in this section, we also believe that mechanisms akin to social affect can play an important role in creating more effective computational systems. For social affect, we are interested in "social" computational systems. This can include both networked systems and single systems with multiple interacting processes, though our initial efforts have focused on applications to networked systems.

In the case of networked computers, we believe that social/herd contagion can provide the basis for a mechanism for rapidly communicating network-based attacks. In this case, if one system detects (or even suspects) an intruder or attack, it can send a rapid, emergency message to all other connected computers warning them to increase security. Such a mechanism would be increasingly useful as cyber attacks become more common and more potentially crippling to U.S. systems. We also believe that social affect can be used to rapidly communicate network needs from one system to others in a way that could cause those other systems to respond and provide additional resources. In an affect-response mechanism akin to sadness and pity, if a computer needs network resources for an important task, but is not getting them, that computer

can alert the other computers who, if not performing similarly critical tasks, can back off of their network usage. Such a mechanism would be especially useful for low-bandwidth systems, such as Ground Soldier systems.

Task 3: Design Affective Reasoning Modules. Instead of fully designing this suite of affect-inspired adaptation mechanisms, we focused our efforts on a computational implementation of the affective attention and resource management module described above, which is inspired by the role that affect (in particular, fear and hope/anticipation) plays in human attentional and motivational focus. Human affect drives the allocation of both mental resources (e.g., attention, memory) and physiological resources (e.g., blood flow, adrenaline) based on ongoing experiences and goals. For example, when humans are anxious, their attention becomes focused on the things causing anxiety and on their safety, at the cost of attention to other actions or events. Conversely, when humans are content, they think more broadly and more effectively perform general problem solving tasks. Ultimately, these changes prepare the anxious person to recognize and overcome or escape the source of the anxiety and prepare the content person for improved learning, and ultimately improved quality of life. In terms of physiology, human affect drives physiological resource allocation, diverting blood flow to hands during bouts of anger and to the chest in the presence of fear (preparing for a potential fight or flight reaction, respectively).

We have extended this concept as our initial module for MARVIN, designing the *Affective Process Management Module* (APMM) illustrated in Figure 2-1. The APMM borrows ideas from human affect-based resource management by monitoring the current situation for all applications, maintaining a table of the *affective* attributes for those applications (e.g., likelihood of success, urgency, cost of failure, benefit of success). The basic APMM architecture is adapted from (Neal Reilly, 1996), where it was used to model the generation and influence of affect in software agents. The architecture is, in turn, based on the cognitive-appraisal theory of emotion put forth in (Ortony et al., 1988).

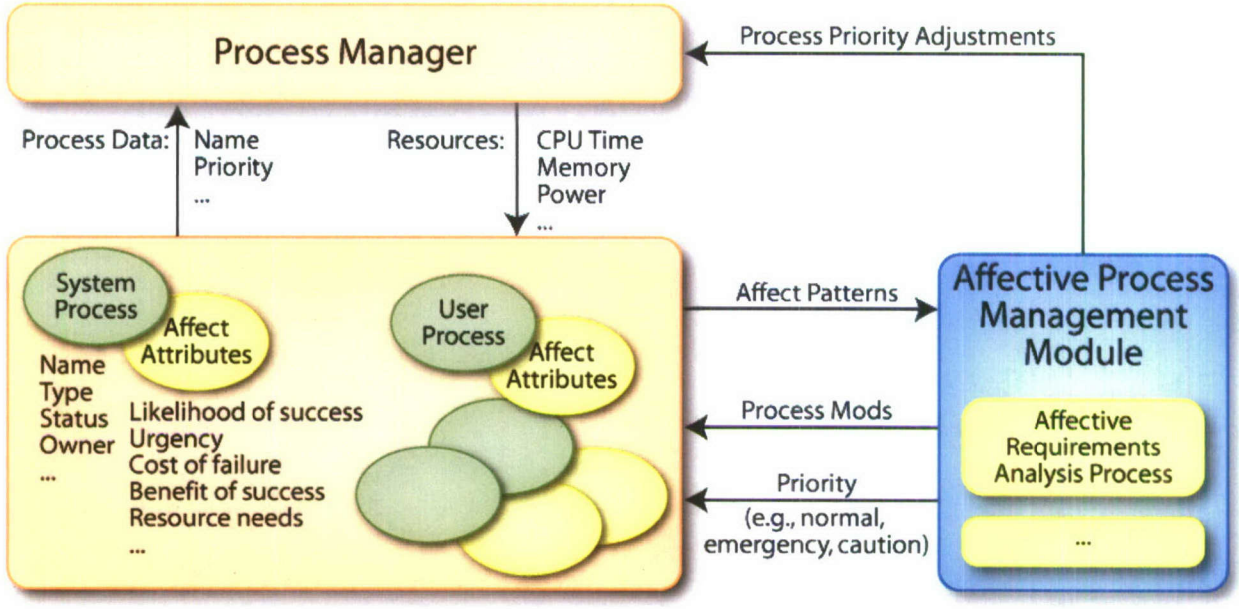

Figure 2-1: APMM Conceptual Design

In this approach, S*ystem Processes* (e.g., operating system threads that correspond to software applications) are extended to support A*ffect Attributes* (e.g., likelihood of success) in addition to standard annotation (e.g., name, status). The *Affective Process Management Module* looks for *Affect Patterns*, which are inspired by the affective literature, that match affective situations such as a lowered likelihood of success for a high-importance process. The patterns correspond to process-specific "emotions." These emotions have a type (e.g., fear), and intensity (which in the case of fear, is based on the importance of the process not failing and the likelihood that it will if not tended to), and the cause of the emotion (e.g., fear of failing due to lack of CPU resources or network bandwidth). The latter is used to provide more specific and appropriate emotional response, just as in humans fear of failing a test and fear of a mugger produce different responses that are suited to the situation (Neal Reilly, 2006). The emotions are used by the Affective Process Management Module to reallocate resources (e.g., CPU) via the standard *Process Manager* for the OS. It can also update the Affective Attributes that can signal the self-aware system processes to change their behavior to, for instance, enter an emergency mode where critical data is saved or sent.

Self-aware applications can be designed for use with the APMM, which enables them to update their affective attributes as needed to improve performance at key junctures. For instance, the self-aware UAV will know that the success of its avoid-obstacles objective is threatened when an obstacle in seen to be in its flight path and the urgency of this threat is based on its distance. Such application-specific knowledge is most easily encoded in the applications themselves, as long as they have mechanisms for communicating that knowledge, which we

provide as methods that change the values of their affective attributes. Those applications that do not use any affective attribute management will execute under their standard priority, with no modifications made by the APMM. When an application updates its affective attributes, the APMM analyzes those updates and determines changes in the application's priority and resource requirements. Based on these changes, the APMM will direct simple adjustments to process priority through the OS Process Manager. Additionally, it can update the affective attributes itslef, potentially changing the way those applications focus whatever resources they receive. Ultimately, the APMM emulates the human affective tendency to assign more resources to resolving "anxieties" (e.g., application emergencies) by focusing attention on those anxieties (e.g., raising the priority of those processes).

Consider a practical example of an affective UAV gathering information in a particular region of airspace. While "content," it fairly distributes resources based on standard process priorities, just as a normal UAV. However, when this affective UAV notices a potential collision, its affective processes react to ensure high priority goals are achieved. In this case, the key goals are to avoid the collision and to ensure that all data collected is downloaded to a network resource. Thus, the affective navigation might report a dramatic increase in urgency which is combined with the high cost of failure associated with this objective not being met. This combination of factors would cause the APMM to significantly increase the priority of the navigation process, which, in turn, would cause the UAV to shift additional resources to that process. In addition, it the APMM would alert the other processes that it is an emergency state, causing the data-transfer routing to attempt to download its data. This sequence of events has three results. First, it can reduce crashes by enabling the navigation system to focus on the threat. Second, in the case where there is a crash, this process can result in the affective UAV reporting more data than a non-affective UAV. Third, in the case that the crash is avoided, the *fear* or *anxiety* would abate, the affective UAV would return to normal operations, and the disruption would be no more than the provision of an early data report.

We believe the APMM provides a simple affect-inspired modification that can be made to any OS with minimal disruption to the operation of the OS kernel. From the perspective of computational resources, this module adds a single simple computational process for calculating affective priority for each application as a function of the importance, urgency, threats, and opportunities associated with each application. This capability can be manipulated and optimized to ensure its execution does not disrupt pre-existing process management requirements. Affect-ready applications are enhanced to update their own affective attributes, which are particular to each application and so cannot be reasonably pulled out into the APMM itself. Furthermore, the APMM and affective processes have a negligible memory footprint, particularly when compared

to modern memory sizes, even on relatively low memory systems, such as Soldier-based PDAs or similar systems. We also believe that even these minimal costs to performance are offset by the fact that the APMM supports a number of key requirements established for organic or autonomic systems. It provides a high-level *self-awareness* of the status of local applications, allowing it to *adaptively* and dynamically reconfigure the allocation of resources to those applications. It supports system *resiliency*, allowing resources to be funneled to threatened processes to avoid process failure. Additionally, it can provide *goal-oriented behavior* by focusing on those processes that are essential to achieving system intentions (as in the UAV example). Finally, because the APMM is a relatively simple enhancement, it is likely to prove much more *non-disruptive* than many other autonomic system components.

Task 4: Develop Demonstration Prototype. We developed a working prototype of the APMM module within the MINIX 3 OS in support of Task 4. We chose the MINIX OS for demonstrating prototype MARVIN components because it is a classic, but simple, open-source OS designed for teaching operating systems to students. MINIX provides a minimum barrier for acquisition, understanding, and modification, and thus provides an ideal test environment for rapidly developing MARVIN modules, especially within the time and budget limitations of a Phase I effort.

As a microkernel architecture, MINIX simplifies the integration of prototype OS features, as those features can be rapidly developed as stand-alone components that are external to the microkernel, avoiding the intensive and often tedious optimization process necessary to integrate such components directly into the kernel (as would be necessary in other open-source mainstream OSs). MARVIN modules are implemented as server processes in MINIX, as illustrated in Figure 2-2. This allows us to rapidly develop and test the advantages of new MARVIN modules, proving their feasibility before undergoing the more intense effort required to transition them to mainstream OSs. However, while this dramatically eases our testing process, it becomes a potential disadvantage when trying to generalize MARVIN modules to monolithic kernels found in most mass market OSs, where these kinds of system processes run within the kernel, and thus need to be highly optimized as well as connected to more integrated kernel elements. We are minimizing this risk by encapsulating MARVIN, which will simplify the integration process and enable us to focus on the optimization process during transition. In Figure 2, the gray, dashed circles and lines indicate existing MINIX OS processes and communication paths, while the blue boxes and solid black lines show how we intend to follow the MINIX model to extend and enhance the OS with affective capabilities.

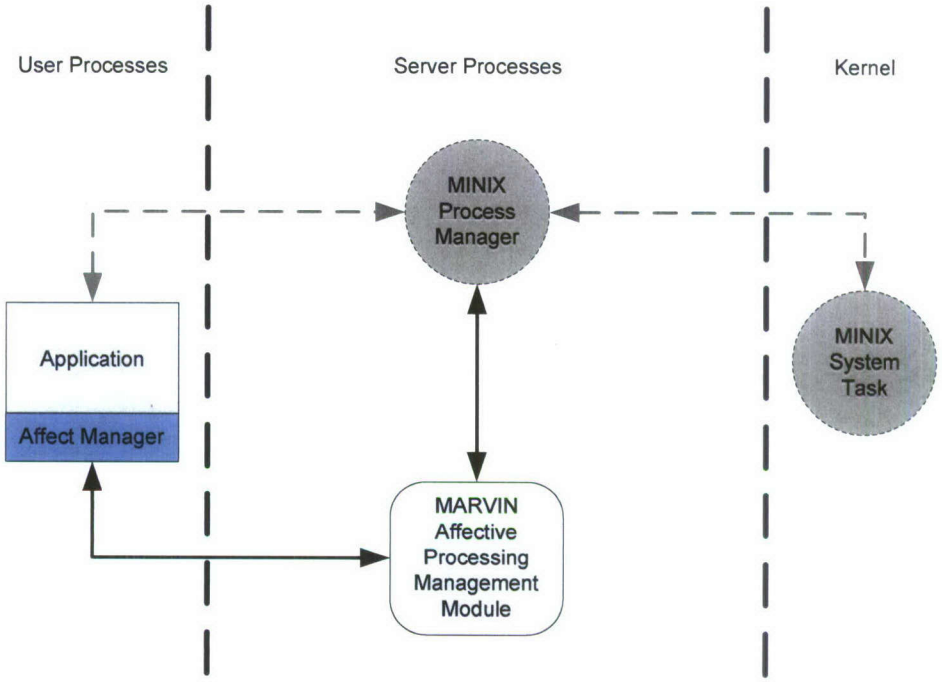

Figure 2-2: MINIX Process Level for APMM Components

To enforce this encapsulation, we have designed the APMM as a new MINIX Server Process, as illustrated in the middle component of Figure 2-3. The APMM's Core Services provide support for two main system features: (1) an `AffectiveCalculationEngine` capable of analyzing processes based on their affective attributes (this component can be potentially expanded to support a selection of calculator based on "mood" or "personality traits" of the OS); (2) an `AffectiveProcessTable` (disjoint from, but in synch with, the primary process table kept by the MINIX Process Manager) that maintains affective attributes for particular processes. Additionally, it is designed to support future implementation of a `RapidResponseQueue` that maintains the list of highest priority processes for rapid access during emergency situations (this component is intended to support an emergency rapid resource allocation aspect of the APMM that will not be included in our initial implementation). Additionally, the APMM is the handler for all calls in a new Affective Interface Library that provides applications with an interface template to enable their communication of affective features to the APMM. Lastly, the APMM implements an interface to the MINIX Process Manager to delegate priority changes to and to stay abreast of state changes to the process in the primary process table. Downstream, we plan to provide a suite of basic affective management algorithms as part of this library to enable the OS user to assign established affective patterns to otherwise non-affective processes.

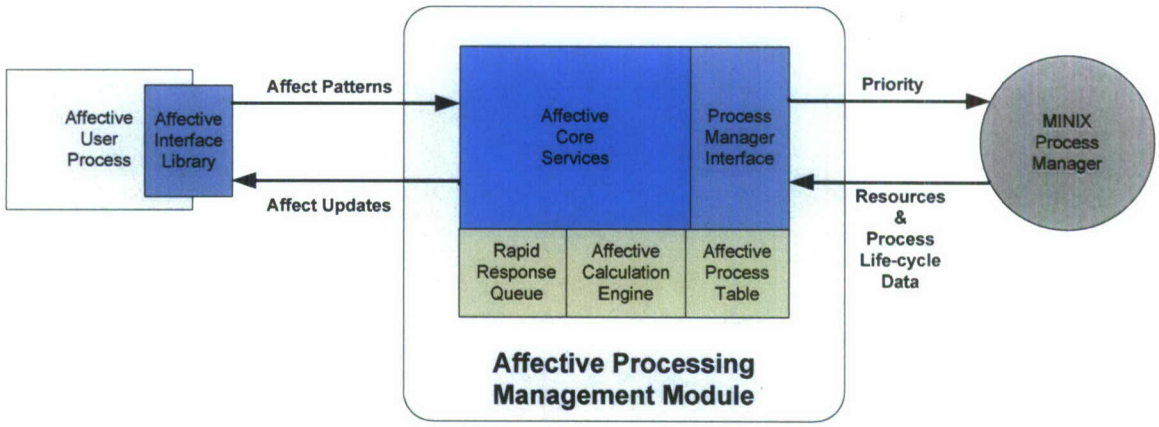

Figure 2-3: APMM Internal Architecture Diagram

Because the APMM encapsulates the extended affective functionality without modifying pre-existing process management elements, we expect the process of transitioning this component from MINIX to monolithic kernels to be straightforward, with the element of process optimization to enable insertion into a kernel overshadowing the complexity of integration into parallel OS components. Even this optimization process should be straightforward for the APMM, as it achieves its advantages with the addition of simple features to process management. As long as we maintain this level of encapsulation, we believe that MINIX will continue to provide a strong testing environment for new MARVIN modules, allowing us to rapidly implement and benchmark new affective modules before performing the software optimization necessary to incorporate them into a monolithic kernel.

To demonstrate how the APMM would work in practice, we used it as the basis for building a simulated UAV controller. The UAV is responsible to collecting and delivering time-critical intelligence, surveillance, and reconnaissance (ISR) data while also navigating to avoid threats and obstacles. It must do these things given a limited CPU and limited time to respond to dynamic events, including ISR requests and obstacle avoidance. The simulated UAV has ten parallel processes that are responsible for tasks relating to Avionics (AV) [1 process], Threat Detection and Avoidance (TDA) [1 process], Sensor (SEN) [4 processes], and Data Transmission (XFR) [4 processes] system processes. These processes were given a scripted (repeatable) sequence of events to respond to, each with a time window for responding.

We built both a non-affective version of this system, where each process is given a suitable priority (e.g., TDA is higher priority than SEN processes) but that priority is static for the duration of the run, and an affective-version of the system, where each process is responsible for recognizing threats to accomplishing the tasks associated with that process and updating its likelihood-of-failure and likelihood-of-success fields based on the time available and the anticipated time required to respond. In the affective case, the APMM updates the "emotional

state" associated with threats to accomplishing objectives that are based on the level of the threat and the importance of the task. These "emotions" in turn give rise to dynamic changes in the priorities of the processes.

We found that the affective system was able to reallocate system resources (in this case, CPU time) to effectively respond to time-critical events. One concern with this approach was that we might have found that this performance required greater overhead and therefore resulted in degraded performance in non-time critical situations. In reality, since the reallocation is dynamic and based on current need, we found that those processes not requiring significant resources were able to use fewer resources, resulting in overall improved system performance for event handling in non-time-critical situations as well. Below, we provide additional details of the tests we performed to evaluate the effectiveness of the APMM implementation and to gather some insights into possible drawbacks of the approach.

Task 5: Develop Evaluation Metrics. It will clearly be important to measure success of self-aware systems against conventional metrics, such as overall CPU time used, time spent swapping, thread count, thread starvation, memory usage, memory access time, context-switching costs, network bandwidth, network latency, and others. These traditional metrics, however, may not capture all of the benefits and tradeoffs of self-aware systems, especially as we try to compare various self-aware systems. For instance, system stability and resiliency (e.g., how effectively does the system stay online in the face of system threats), efficiency at achieving important system goals, important-traffic throughput/latency are all important metrics and ones we expect self-aware systems to excel at, though none are typically used to measure system performance.

Based on our effort to focus less broadly and more deeply on the overall Phase I effort, we did not spend significant time developing general-purpose evaluation metrics, but rather focused on developing and implementing a number of evaluations for the affective resource manager that we developed under Task 4. So, under this task, we attempted to identify a number of metrics for the affect-based resource allocation mechanism as applied to the simulated UAV controller. We also attempted to gather data against those metrics. While we did not attempt to generate a complete list of appropriate metrics for this domain, we did attempt to choose a small number of useful metrics for demonstration purposes.

First, we instrumented the code for the affective and non-affective processes and measured the time that they required to respond to key environmental events. Figure 2-4 shows these times over the course of the UAV simulation. We see that the times vary for both the affective and non-affective processes, based in part on the other concurrent processes, but that the affective

processes typically outperform the non-affective processes by responding to critical events roughly 80% faster.

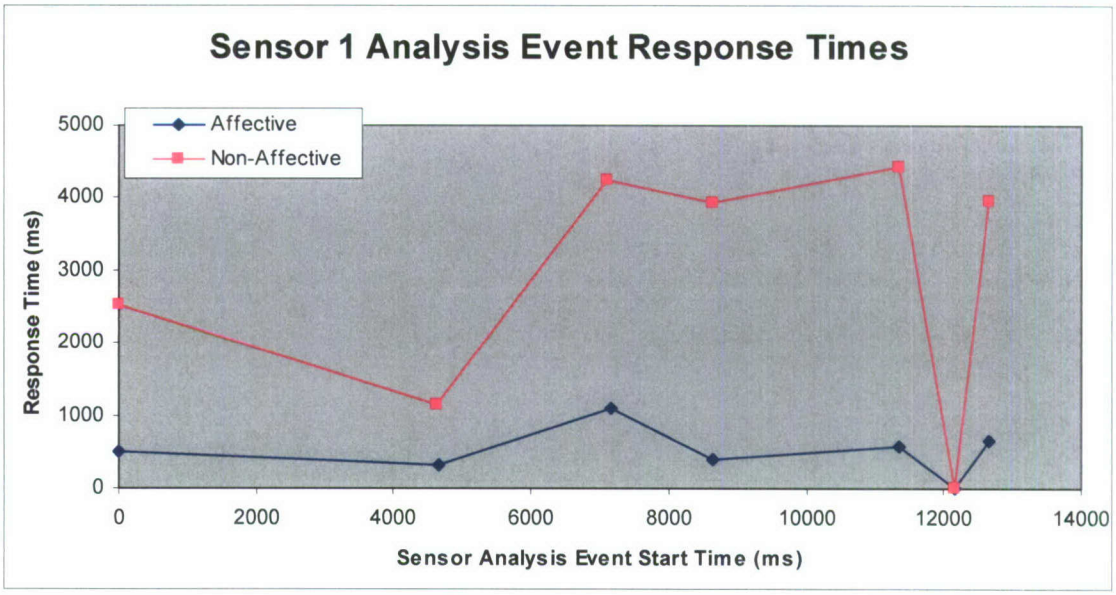

Figure 2-4. Process Response Times for Critical Affective and Non-Affective Processes

The second evaluation was a broad measurement of the survivability and operational success of the simulated UAV. In this case, we identified a number of time-critical events, some related to data gathering and some related to crash avoidance and measured the ability of the affective and non-affective systems to perform these tasks. We found that the non-affective system was able to respond to 11 of the 60 (18%) time-critical tasks in the scenario and none of the crash-avoidance tasks, while the affective system was able to respond to 38 of the 60 (63%) time-critical tasks and both crash-avoidance tasks. These results are summarized graphically in Figure 2-5.

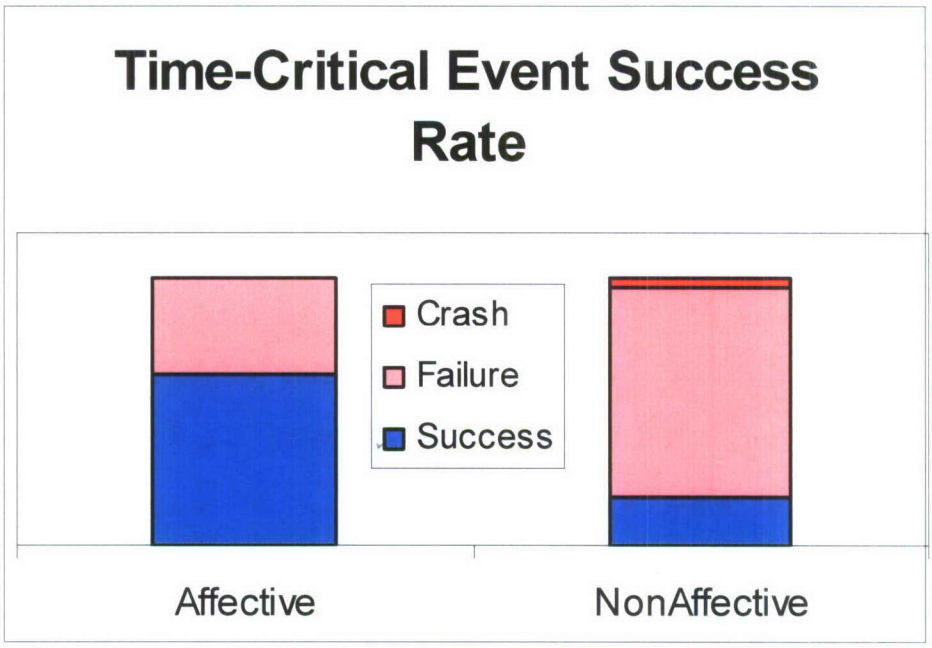

Figure 2-5. Affective vs. Non-Affective Performance on Time-Critical Responses

Finally, we looked at the profile of the actual events that were responded to in both cases to see if there were particular cases where the non-affective system out-performed the affective system. In particular, not all events are time-critical and not all processes high-priority. There was some concern that the overhead of the APMM might reduce performance in these cases. In fact, we found that of the 10 different types of events and responses handled by our simulated system, the affective system performed better on all of them and, in fact, the set of events handled in the allotted time by the affective system was a superset of the events handled by the non-affective system. Figure 2-6 provides evidence for this claim, though additional evidence is in the form of the complete set of event-handling process charts like that in Figure 2-4, where it becomes clear that the full set of processes are more efficient in the affective system and across the full set of events (some time-critical, some not).

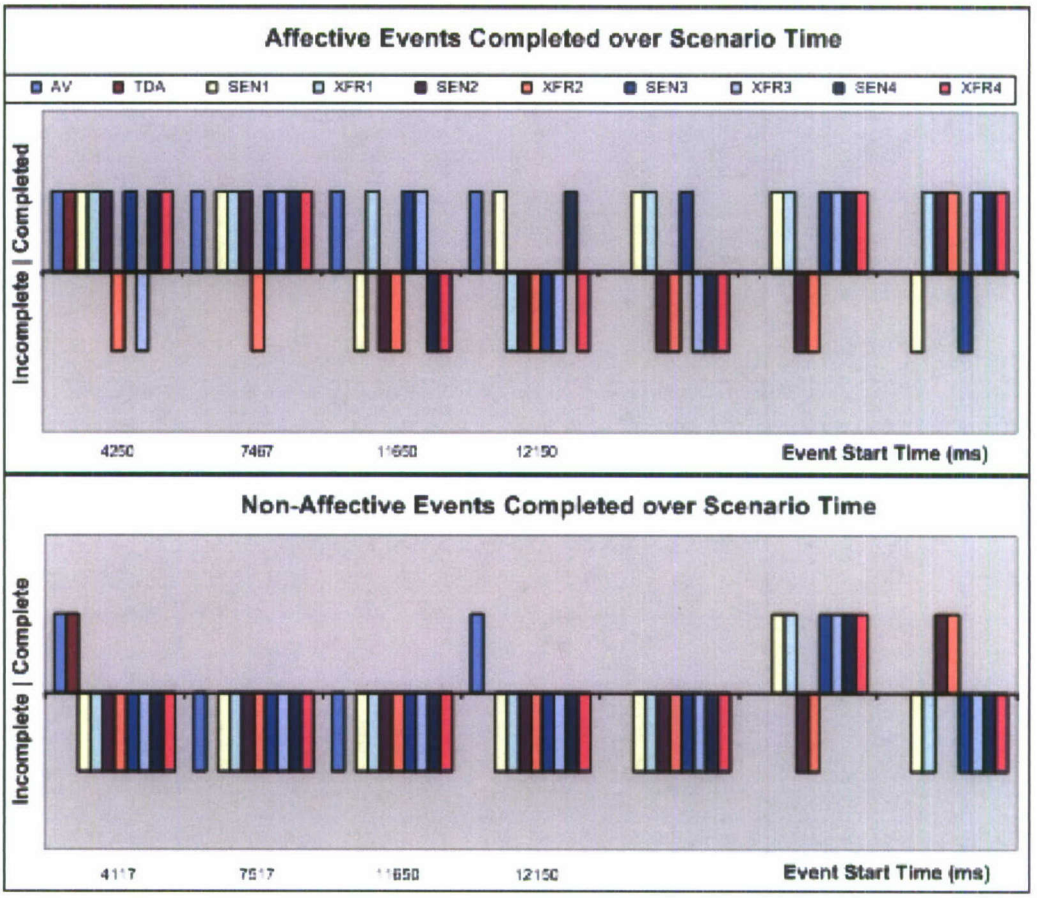

Figure 2-6. Events by type handled successfully and unsuccessfully by the affective and non-affective systems

This is not by any means meant to be a complete evaluation of the affective resource manager in general or even the implementation for this particular application scenario, but it does demonstrate that the basic approach is significantly more effective than the traditional approach to OS resource allocation. It also provides some supporting data that demonstrates that the approach is not worse than the traditional approach in any of the cases we tested it for, which one might believe would be the case if we were simply trading off success in some tasks for failures in others.

Task 6: Investigate Commercialization Opportunities. In order to ensure that the results of this effort are practical and transitionable to operational systems, we have begun identifying particular applications and partners that would be able to support transition efforts during a potential Phase II effort.

We believe significant benefits from the use of these technologies can be seen in unmanned vehicles, such as mini and micro UAVs, and Soldier-based systems, such as the Army's Future

Warrior Technology Integration (FWTI) Team's PDA-based Basic Soldier system and the Thermite- and laptop-based Leader Systems. FWTI is follow-on from the earlier Future Force Warrior (FFW) Advanced Technology Demonstration (ATD). Because of the resource limitations (memory, CPU, network, battery) and time limitations placed on these systems by their use in dynamic, dangerous environments, we believe they are well suited as target platforms. In addition, our participation as a member of the FWIT under a parallel SBIR effort provides us access to the developers, a number of end users, the platform requirements, the platforms, and other contractors that are working on relevant problems.

We have also begun initial discussions with large system integrators that are currently developing software systems for the U.S. Army's FWTI program, a variety of UAVs, and other military robotic systems. Such systems often have limited resources, which we believe make them especially well suited for the kinds of affect-inspired modifications that we are developing under this program. Our discussions on this front have shown some interest from the developers of such systems in exploring the evaluation and eventual transition of MARVIN concepts in operational systems.

3. Conclusions

Under the Phase I effort, we demonstrated that affect-inspired, self-aware mechanisms can significantly improve the performance of computational systems in cases of limited resources. We designed four affect-inspired adaptation mechanisms: an affective attention and resource management module, a generic, multi-tiered affective adaptation module, an emergency resource-allocation module, and a social-affect module. We believe all of these mechanisms have the ability to positively impact the performance of various types of computational systems and identified some such systems for each mechanism.

Based on direction from the Sponsor, instead of refining and extending these designs, we instead focused on developing a mature MINIX-based implementation of a version of the affective attention and resource management module called the Affective Process Management Module (APMM). We developed a reasonably rich simulated UAV-based scenario that required the UAV to respond rapidly to a range of different environmental events. We also implemented a non-affective version of the same system for comparison. Based on our evaluation of the performance of each system in responding to the time-critical events, we found that the affective system dramatically outperformed the non-affective system. This was not, however, at the expense of non-critical operations as the critical response processes were able to dynamically reduce their resource usage in non-critical periods, enabling the other processes to be even more efficient than their non-affective counterparts.

Overall, we believe that we have accomplished the primary objective of this effort, which was to demonstrate the feasibility of identifying and implementing affect-inspired mechanisms to provide improved performance in computational systems. There obviously remains much work to be done, both in the transition of the APMM to a more realistic OS (e.g., using Linux instead of MINIX and operating in a non-simulated environment), the identification of additional affect-inspired adaptation mechanisms, and the refinement and implementation of the three mechanisms that were identified but not implemented in this effort. We expand on these next steps and propose a concrete set of recommendations for future work in the following section.

4. Recommendations for Future Work

We believe these results provide the foundation for additional research and development that focuses on:

- Designing and developing additional affect-based computational models. During Phase I, we designed a number of concepts for affect-inspired, self-aware mechanisms for improving various aspects of performance of computational systems. During, Phase II, we recommend expanding these concepts into full designs and implementing them for the purpose of further evaluating their potential payoff in operational systems. In particular, we recommend expanding on the following concepts:
    - The *Affective Attention and Resource Manager Module,* which we demonstrated by implementing as the APMM and performed initial evaluations of, has shown its basic usefulness but still requires additional refinement. In particular, we focused on allocation of CPU time in the current effort, but processes can "fear" due to other resource limitations, such as lack of network bandwidth or lack of memory, and we want similar mechanisms to support the runtime reallocation of these resources as well.
    - *An Emergency Resource Allocation Module* based on the human primary-appraisal mechanism that provides a secondary operating system process queue to support rapid context switching to critical processes in emergency situations.
    - *A Generic, Multi-Tiered Adaptation Module* based on the adaptation of affective states at multiple temporal resolutions that provides the ability to adapt to a variety of dynamic elements of the environment.
    - A *Social Affect Module* based on emotional contagion that provides rapid, system- or network-wide adaptation to problems, such as network viruses or intruders.
- Rigorously testing and evaluating the performance and resilience improvements provided by these affect-based modules. The evaluations we performed on the APMM are an initial effort in this direction, but even that module requires additional evaluation to quantify its benefits, understand where it provides the greatest benefits, and identify any potential tradeoffs involved in using such an approach. The other modules would need to be similarly evaluated.
- Transitioning the prototype models, both the APMM built in Phase I and other successful prototypes built during the Phase II effort, to the Linux deployment environment. In Phase I, we have used MINIX as it provides a simple way to prototype and evaluate affective modules, but it is not a deployable solution. We believe the next natural step is the Linux operating system as it is open source but still widely deployed, including on some of the

systems that we are most interested in, including the PDA being used by the FWTI team for their Ground Soldier system.

- Exploring government transition and commercialization opportunities. Once we have developed the proven affect modules on Linux, we will be able to begin the transition process to operational systems. During Phase I, we identified a number of such systems and identified suitable partners to support this transition to operational systems. As a next step, we recommend testing suitable affect modules on the FWTI Soldier system or on a suitable UAV platform.

5. References

Agarwal, A. & Harrod, W. (2006). *Organic Computing.* Cambridge, MA: MIT CSAIL / Darpa IPTO.

Anderson, M. L. & Perlis, D. R. (2005). Logic, Self-Awareness, and Self-Improvement: The Metacognitive Loop and the Problem of Brittleness. *Journal of Logic and Computation,* 15(1), 21-40.

Damasio, A. R. (1994). *Descrates' Error: Emotion, Reason and the Human Brain.* New York: G.P. Putnam's Sons.

Ganek, A. G. & Corbi, T. A. (2003). The Dawning of the Autonomic Computing Era. *IBM Systems Journal,* 42(1), 5-18.

Gelenbe, E. & Loukas, G. (2007). A Self-Aware Approach to Denial of Service Defence. *Computer Networks,* 51(5), 1299-1314.

Minsky, M. (2006). *The Emotion Machine.* New York: Simon & Schuster.

Mowbray, M. & Bronstein, A. (2002). *What Kind of Self-Aware Systems Does the Grid Need?* (Rep. No. HPL-2002-266R1). Palo Alto, CA: HP Labs.

Neal Reilly, W. S. (2006). Modeling What Happens Between Emotional Antecedents and Emotional Consequents. In R. Trappl (Ed.), *Cybernetics and Systems 2006.* Vienna, Austria.

Patterson, D. A., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J. et al. (2002). *Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies.* (Rep. No. Technical Report UCB // CSD-02-1175). Berkeley, CA: U.C. Berkeley.

Pfeifer, R. (1993). The New Age of the Fungus Eater. In *Proceedings of Second European Conference on Artificial Life (ECAL).* Brussels.

Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., & Kubiatowicz, J. (2003). Pond: the OceanStore Prototype. In *Proceedings of 2nd USENIX Conference on File and Storage Technologies (FAST '03).* San Francisco, CA.

Sloman, A. & Croucher, M. (1981). Why Robots Will Have Emotions. In *Proceedings of 7th International Joint Conference on Artificial Intelligence.* Vancouver.

Toda, M. (1962). The Design of the Fungus Eater: A Model of Human Behavior in an Unsophisticated Environment. *Behavioral Science,* 7.

US Army. (2006). Future Force Warrior. Soldier Dominance Through Innovation. US Army Natick Soldier Center.